The claims:

1. A method of developing a computer program using bi-directional programming means, the method including:

(1) utilising a visual representation of the program that can be parsed and edited; and,

(2) utilising a syntax code representation of the program that can be parsed and edited;

(3) converting between the visual representation and the syntax code representation by converting the visual representation and the syntax code representation into byte-code representations and comparing the byte-code representations;

wherein, edits in the visual representation are reflected in the syntax code representation, and vice versa, and the bi-directional programming means can be used to build back-end logic for the computer program.

2. The method as claimed in claim 1, wherein the visual representation is a flowchart diagram, structure diagram, work-flow diagram, parse tree or the like.

3. The method as claimed in claim 2, wherein the visual representation includes an extended flowchart element construct.

4. The method as claimed in claim 3, wherein the extended flowchart element construct includes at least:

a callable statement mapping to a syntax language function call;

a condition type clause having a condition part; and,

a variable.

5. The method as claimed in claim 4, wherein other flowchart element constructs are provided and conform to predefined rules.

6. The method as claimed in any one of claims 1 to 5, wherein a parsing algorithm is used to verify the visual representation or the syntax code representation prior to conversion to byte-code representations.

7. The method as claimed in any one of claims 1 to 6, wherein edits in the visual representation and the syntax code representation can be performed iteratively.

8. The method as claimed in any one of claims 1 to 7, wherein the bi-directional programming means can also be used to build a graphical user interface for the computer program.

9. The method as claimed in any one of claims 1 to 8, wherein edits in the visual representation are automatically reflected in the syntax code representation, and vice versa.

10. The method as claimed in any one of claims 1 to 9, wherein pre-existing syntax code can be read and converted into a visual representation.

11. The method as claimed in any one of claims 1 to 10, wherein at any stage of development the byte-code common to the visual representation and the syntax code representation can be compiled into machine level code.

12. A system for providing bi-directional programming means for developing a computer program, the system characterised by:

(1) a visual representation of the program that can be parsed and edited;

(2) a syntax code representation of the program that can be parsed and edited;

(3) a processor to convert the visual representation to a byte-code representation and then convert the byte-code representation to the syntax code representation, or, to convert the syntax code representation to a byte-code representation and then convert the byte-code representation to the visual representation;

wherein, edits in the visual representation are reflected in the syntax code representation, and vice versa, and the bi-directional programming means can be used to build back-end logic for the computer program.

13. The system as claimed in claim 12, wherein the visual representation contains an extended flowchart element construct which includes primary native language semantics selected from the set of:

a callable statement mapping to a syntax language function call;

any other statement mapping to a syntax language assignment statement;

a condition type clause having a condition part;

a compound statement;

error or exception handling; and/or,

one or more variables.

14. The system as claimed in claim 12, wherein there is additionally provided a parsing algorithm to verify the visual representation or the syntax code representation prior to conversion to byte-code representations.

15. The system as claimed in any one of the claims 12 to 14, wherein the byte-code language includes constructs selected from the following set: Assignment; Method Call; If Expression; If/Else Expression; For Loop; Repeat Clause; Do/While Clause; Switch/Case Expression; Synchronized; Try/Catch/Finally; End Block; or any other higher level language constructs.

16. The system as claimed in claim 12, wherein the visual representation is a flowchart diagram, structure diagram, work-flow diagram, parse tree or the like.

17. A computer program product for use in developing an application program, said computer program product providing bi-directional programming means and comprising:

(1) means to display a visual representation of the program that can be parsed and edited;

(2) means to display a syntax code representation of the program that can be parsed and edited;

(3) means to convert between the visual representation and the syntax code representation by converting the visual representation and the syntax code representation into byte-code representations and comparing the byte-code representations;

wherein, edits in the visual representation are reflected in the syntax code representation, and vice versa, and the computer program product can be used to build back-end logic for the application program.

18. The computer program product as claimed in claim 17, wherein the visual representation includes an extended flowchart element construct.

19. The computer program product as claimed in claim 17 or claim 18, wherein the visual representation contains an extended element construct and includes primary native language semantics selected from the set of:

a callable statement mapping to a syntax language function call;
any other statement mapping to a syntax language assignment statement;
a condition type clause having a condition part;
a compound statement;
error or exception handling; and/or,
one or more variables.

20. The computer program product as claimed in claim 19, wherein the extended element construct contains a generic data structure for the visual representation including a start element, an end element and a means of representing a symbol table.

21. The computer program product as claimed in claim 19, wherein the extended element construct contains a generic data structure for the representation of a collection of symbols or a symbol table.

22. The computer program product as claimed in claim 19, wherein the extended element construct contains a generic data structure for the representation of an individual symbol.

23. The computer program product as claimed in claim 19, wherein the extended element construct contains a generic data structure for the representation

of an individual node of the visual representation.

24. The computer program product as claimed in any one of the claims 19 to 23, wherein the extended element construct is an extended flowchart diagram, structure diagram, work-flow diagram or parse tree element construct.